Übungsbeispiele – PR1

Beispiel 1 – TurmRechnen

Erstellen Sie eine neue Klasse mit dem Namen „TurmDemo inkl. main-Methode. Das Programm soll für eine von Ihnen festgelegte Zahl die klassische Turmrechnung durchführen:

2 * 2 = 4

4 * 3 = 12

12 * 4 = 48 usw.

Tipp: Schreiben Sie zwei while-Schleifen, welche jeweils von 2 bis einschließlich 9 laufen sollen. Multiplizieren bzw. Dividieren Sie jeweils das Ergebnis aus dem Durchlauf davor mit dem aktuellen Wert des Schleifendurchlaufs.

Beispiel 2 – Grenzwert Ermittlung

Erstellen Sie die **Klasse GrenzwertReihe mit main-Methode().** Schreiben Sie unter Zuhilfenahme von Schleifen und Bedingungen ein Programm, das für eine bestimmte Genauigkeit den Grenzwert der folgenden Reihe ausgibt.

$$\sum_{n=0}^{\infty} \frac{1}{2^n} = \frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots$$

Dazu soll die Methode

```
public static double berechneGrenzwert(double genauigkeit)
```

implementiert werden. Rufen Sie diese Methode dann aus main auf, um sich den Grenzwert für 3 versch. Genauigkeiten (0.01 , 0.001, 0.0001) zu ermitteln und geben Sie diesen auf der Konsole aus.

Hinweise:

Ein Grenzwert ist gefunden, sobald sich der **aktuelle Wert (aWert)** der Reihe zum unmittelbar **vorhergehenden Wert (vWert)** nicht mehr bzw. **nur mehr geringfügig verändert**. Dies kann z.B. mit folgender Bedingung ermittelt werden → **if(aWert – vWert < genauigkeit)**

Sofern diese Bedingung **true** ist können Sie mit **break** die Schleife abbrechen. D.h. Sie haben hier keine fixe Abbruchbedingung und der Termination-Ausdruck der for-Schleife bleibt ungenützt.

Beispiel 3 – Verschachtelung von Schleifen:

Erstellen Sie die **Klasse KleinesEinMalEins mit main-Methode().** Implementieren Sie **mittels 2 geschachtelter for-Schleifen** die gesamte Ausgabe auf der Konsole für das kleine EinMalEins:

```
 1     2     3     4     5     6     7     8     9     10
 2     4     6     8     10    12    14    16    18    20
 3     6     9     12    15    18    21    24    27    30
 4     8     12    16    20    24    28    32    36    40
 5     10    15    20    25    30    35    40    45    50
 6     12    18    24    30    36    42    48    54    60
 7     14    21    28    35    42    49    56    63    70
 8     16    24    32    40    48    56    64    72    80
 9     18    27    36    45    54    63    72    81    90
 10    20    30    40    50    60    70    80    90    100
```

TIPP: Verwenden Sie in den **System.out.print(…)** Statements das **Zeichen '\t'** wodurch ein **Tabulator** ausgegeben wird und die Ausgabe schöner als mit Leerzeichen ausgerichtet werden kann!

Beispiel 4 – Arrays:

Erstellen Sie eine neue Klasse namens “ArraysDemo” inkl. main-Methode. Initialisieren Sie ein beliebiges Ganzzahlen-Array.

a) Schreiben Sie eine statische öffentliche Methode mit folgender Signatur:

```
public static boolean containsNumber(int[] numbers, int n) {…}
```

Diese Methode soll feststellen, ob im übergebenen Array numbers die Zahl n vorkommt. Falls ja wird true ansonsten false zurückgeliefert.

b) Schreiben Sie eine statische öffentliche Methode mit folgender Signatur:

```
public static int calculateSum(int[] numbers) {…}
```

Die Methode soll die Summe der im Array befindlichen Elemente berechnen und diese zurückliefern.

c) Schreiben Sie eine statische öffentliche Methode mit folgender Signatur:

```
public static int[] reverse(int[] original) {…}
```

Die Methode soll in der Lage sein, ein neues Array zu erstellen. In diesem sollen sich die selben Zahlen, wie jene im übergebenen Array befinden, jedoch in umgekehrter Reihenfolge. Das resultierende Array soll zurückgeliefert werden.

d) Schreiben Sie eine statische öffentliche Methode mit folgender Signatur:

```
public static double[] calcStats(double[] input) {…}
```

Die Methode soll in der Lage sein, ein neues Array zu erstellen. In diesem sollen die folgenden 3 Werte gespeichert werden: auf Index 0 die kleinste Zahl, auf Index 1 die größte Zahl und auf Index 2 der Durchschnittswert. Das resultierende Array soll zurückgeliefert werden.